

Lecture 3: Fortran 90 (part I)

1. Structure of programs
2. Data types
3. Operators
4. Arrays
5. Input / output

Structure of Fortran 90 programs

A typical Fortran 90 program consists of

- **one main program**

```
program [name]
  [use module_name]
  [implicit none]
  ! declarations of variables
  ! initialization of variables
  ! assignments, function calls, and subroutine calls

end program
```

- **possibly several functions and subroutines**

```
subroutine name(arg1[,args])
  [implicit none]

end subroutine name

function fun(arg1[,args])
  [implicit none]

end function fun
```

- **possibly a number of modules**

```
begin module module_name
  ! declaration of variables

contains !subroutines and function may follow

end module module_name
```

Data types

Real numbers:

```
real(4):: a, var           ! 4-byte real numbers; real*4
real(8):: ca, g, fun       ! 8-byte real numbers; real*8

a=8.0
var=10.5*exp(a)
g=3.4d0**3                 ! can use 3.4 instead of 3.4d0
ca=2.d-4/sqrt(g)          ! can also use 2.4e-4 instead of 2.4d-4
```

Integers:

```
integer:: i                ! standard 4-byte integer
integer(8):: num, mlong    ! long 8-byte integer

i=2
num=10 !assigns num=int(10,8)
mlong=num**12              ! cannot be a 4-byte integer!
```

Complex numbers:

```
complex:: z1               !consists of a pair of real(4) numbers
complex(8):: zz           !pair of real(8) numbers; complex*16

z1=(8.2,2.6e-3)           ! (real part, imaginary part)
zz=(34.d0,2.9e1)
```

Data types (cont'd)

Logical:

```
logical:: done, answer, dd
done= .true.
answer=.false.
dd = .not. answer .and. done
```

Characters:

```
character(len=20):: string

string='fortran is fun' !up to 20 characters fit here
write(*,*) string
```

Derived data types ! Your type definition.;like structures in C

```
type student
    character(len=20) first_name, last_name
    integer:: id
end type student
```

```
type(student):: buzz, copy
buzz%first_name = 'peter'
buzz%last_name = 'smith'
buzz%id = 5498527
copy = buzz
```

Operators

numeric (integer, real, complex) : increasing order

+ ! addition operator: $c = a+b$

- ! subtraction: $c = a-b$

+ ! sign: $a = +4$

- ! sign: $a = -3.4$

* ! multiplication: $c = a*b$

/ ! division: $c = a/b$

** ! power :: $a = b**2$

Logical (decreasing order):

.not. .and. .or.

Relational:

.eq. .ne. .lt. .le. .gt. .ge.

== /= < <= >= >=

Arrays

Arrays are one of the most useful data structures in computational physics!

Arrays are stored in columns.

Declarations:

Example 1

```
real(4):: a(100), vec1(27), mat1(10,32)
real(4), dimension(27):: vec2
real(4), dimension(10,32):: mat2
```

Example 2

```
integer, parameter:: n=20           !fixes n once and for all
real(4):: b(n), cc(n,n)             !declares a vector and a matrix
```

Example 3

```
real(4):: a(10)
real(4), allocatable, dimension(:):: v1, v2           !declares two vectors
real(4), allocatable, dimension(:,):: m1, m2         !declares two matrices

allocate(v1(10), v2(size(a)), m1(3,6), m2(6,8))      !allocates memory

deallocate(v1,v2,m1,m2) !free memory once arrays are not used anymore
```

Array arithmetic

Array constants:

```
integer, dimension(4), parameter :: values=(/ 2, 4, 87, 34 /)
```

Array assignments:

```
real(8) :: v(3), c(3,2), mat(2,2), a(2), num
v(1) = 1.2
v(2) = 43.8
v(3) = 2.0
c(1,1)=1.1
c(2,1)=2.0
c(3,1)=4.0
c(1:3,2)= v(1:3)    ! this initializes the second column
a      = v(2:3)     ! this assigns elements 2 and 3 from v to a
```

Vector and matrix operations:

```
mat = matmul(transpose(c),c) !matrix multiply yields 2 x 2 matrix
v   = matmul(c,a) !matrix vector product yields vector
num = dot_product(v,v) ! dot product of two vectors
```

